MATH 598: TOPICS IN STATISTICS

BAYESIAN NONPARAMETRIC METHODS

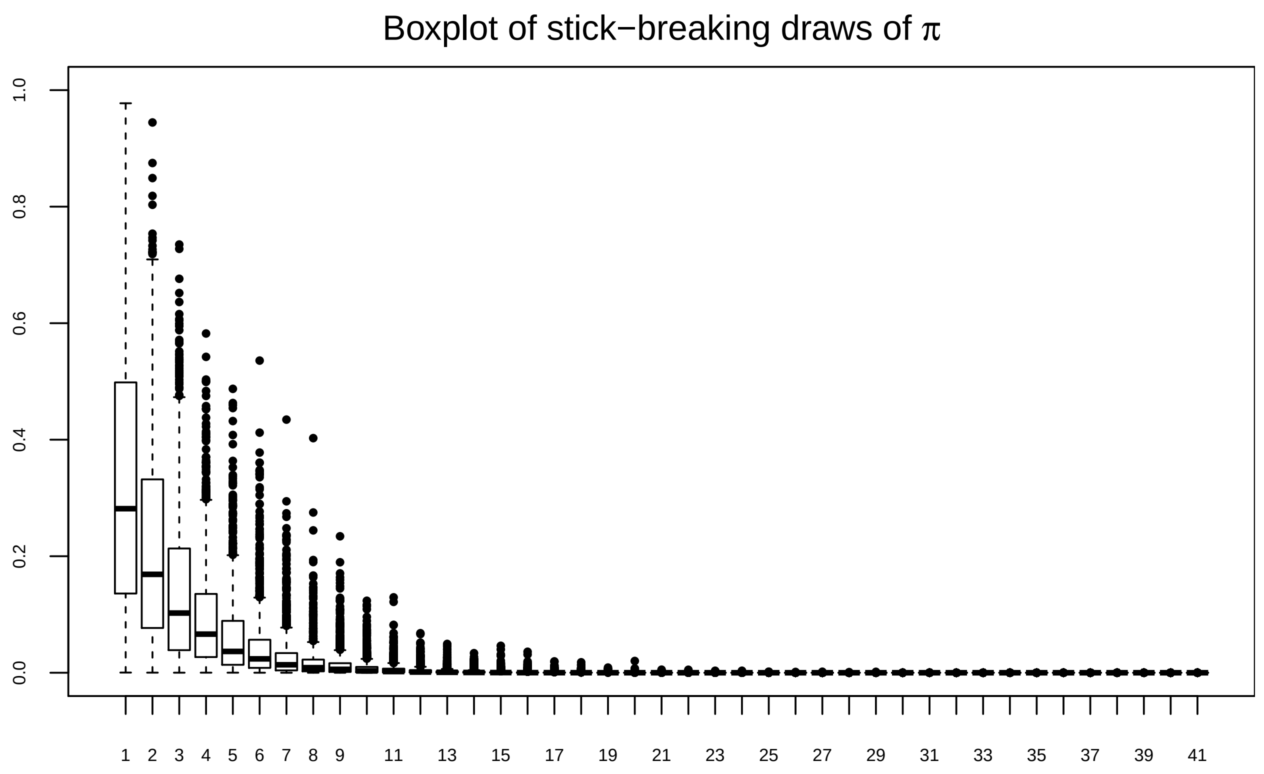
The *Dirichlet Process* is a probability distribution on the space of discrete distributions on $\mathbb{R}$ characterized by

- a base measure, G_X, which is some probability measure on $\mathbb{R}$, from which are drawn a countable collection of random variables $X_1, X_2, \ldots$.
- a countable collection of random variables $\pi_1, \pi_2, \ldots$ with $\pi_i > 0$ such that $\sum\limits_{i=1}^{\infty} \pi_i = 1$ that is constructed in the following way: $\pi_1 = V_1 \sim Beta(1, \alpha)$ and

$$\pi_i = V_i \prod_{j=1}^{i-1} (1 - V_j)$$

where for each j, $V_j \sim Beta(1, \alpha)$ for some parameter $\alpha > 0$. Note that by construction, in expectation $\pi_1 > \pi_2 > \cdots$, which facilitates truncation at some finite value M.

```r
M<-41
alpha<-2
pi.mat<-matrix(0,nrow=1000,ncol=M)
for(i in 1:1000){
    V<-rbeta(M,1,alpha)
    pi.mat[i,]<-c(V[1],cumprod(1-V[-M])*V[-1])
}
par(mar=c(2,2,3,0))
boxplot(pi.mat,pch=19,cex=0.5,ylim=range(0,1),cex.axis=0.6)
title(expression(paste('Boxplot of stick-breaking draws of ', pi)),line=1)
```

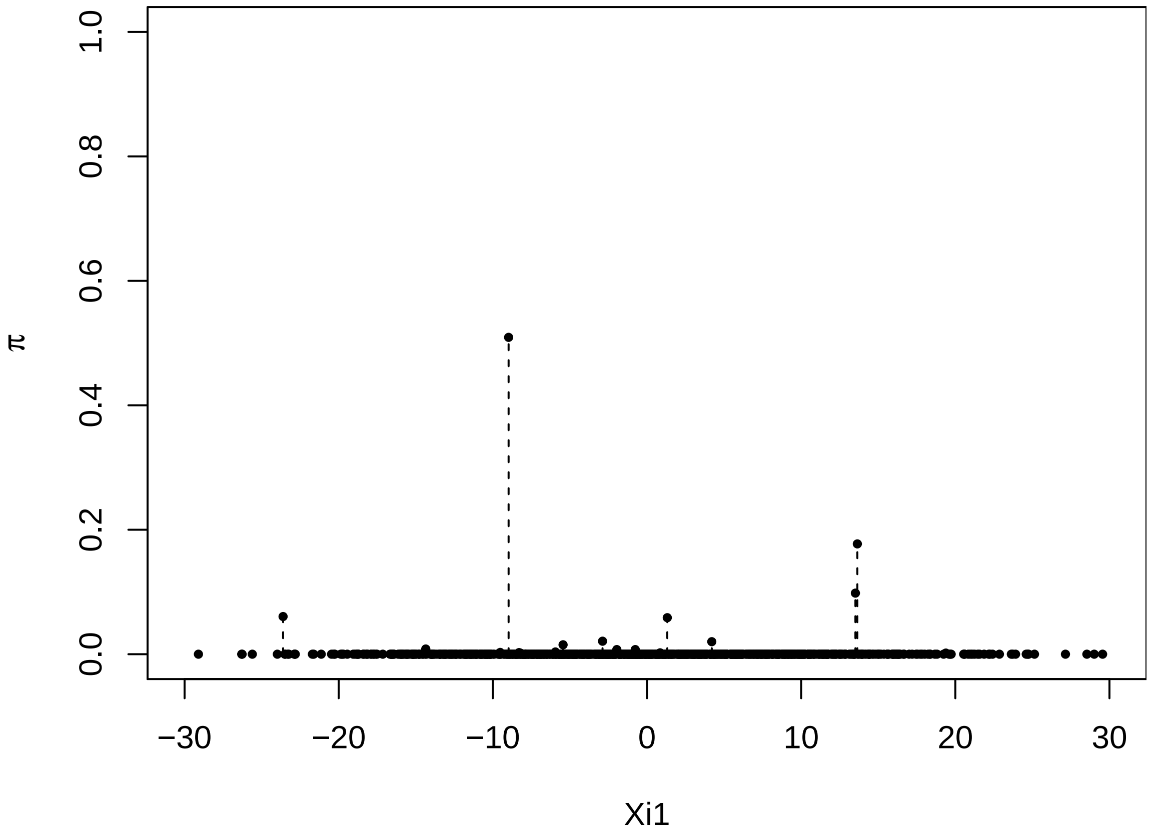


The collections $\{X_1, X_2, \ldots\}$ and $\{\pi_1, \pi_2, \ldots, \}$ define a random discrete distribution with mass function $\widetilde{f}$ where

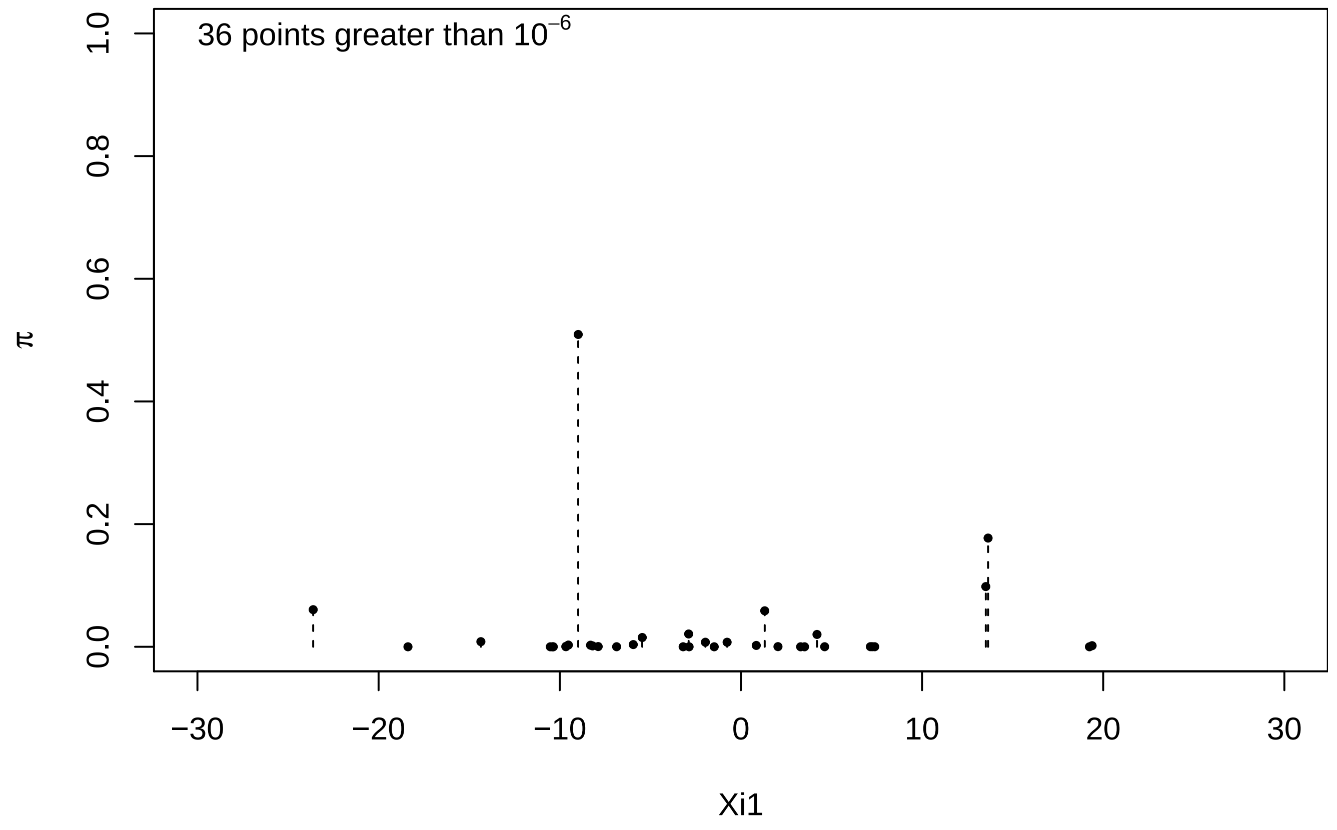
$$\widetilde{f}(x) = \sum_{i=1}^{\infty} \pi_i \delta_{X_i}(x)$$

where $\delta_{x_i}(x) = 1 \iff x = x_i$. We write $\widetilde{f} \sim DP(\alpha, G_X)$ to denote that $\widetilde{f}$ is drawn from the Dirichlet Process.

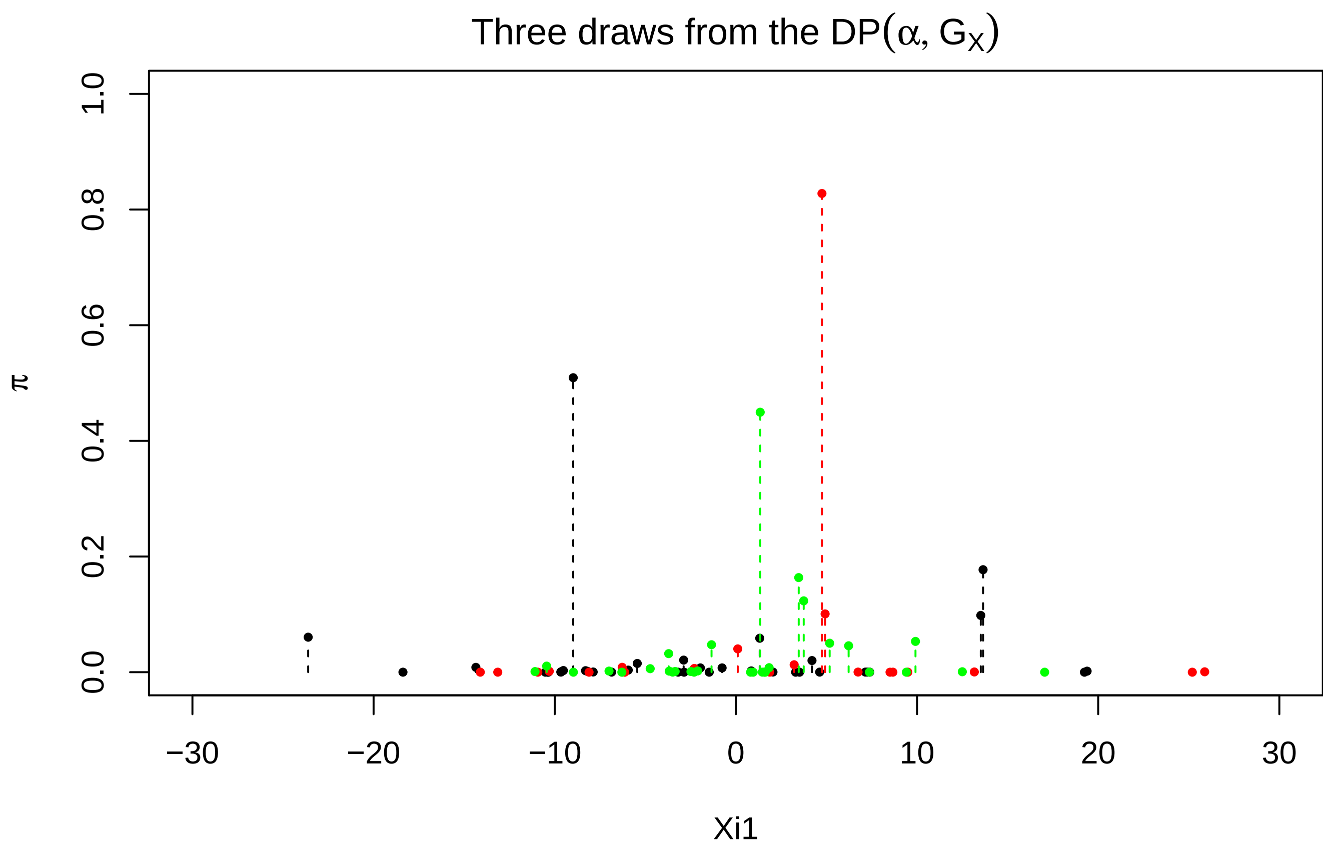
```
M<-1001
alpha<-2
set.seed(1010)
#Stick-breaking
Xi<-rnorm(M,0,10)   #Base measure
V<-rbeta(M,1,alpha)
pi.vec<-c(V[1],cumprod(1-V[-M])*V[-1])
Xi1<-Xi
pi1<-pi.vec
par(mar=c(4,4,1,0))
plot(Xi1,pi1,type="n",ylim=range(0,1),ylab=expression(pi),xlim=range(-30,30))
points(Xi1,pi1,pch=19,xlim=range(-30,30),cex=0.5)
for(i in 1:M){lines(c(Xi1[i],Xi1[i]),c(0,pi1[i]),lty=2)}
```



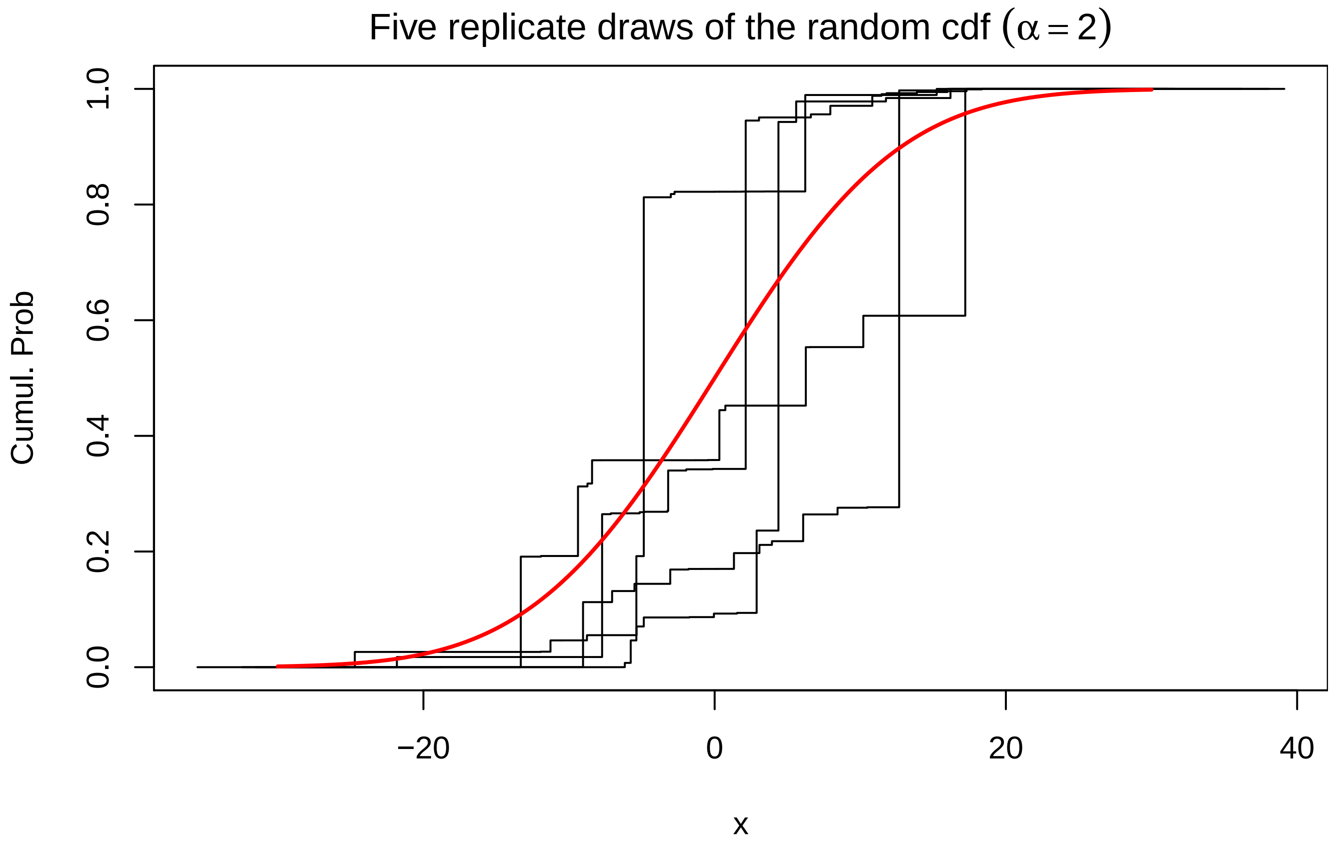
```
par(mar=c(4,4,1,0))
plot(Xi1,pi1,type="n",ylim=range(0,1),ylab=expression(pi),xlim=range(-30,30))
points(Xi[pi1>1e-6],pi.vec[pi1>1e-6],pch=19,xlim=range(-30,30),cex=0.5)
for(i in 1:M){
        if(pi1[i] < 1e-6) next
        lines(c(Xi1[i],Xi1[i]),c(0,pi1[i]),lty=2)
}
nval<-sum(pi1>1e-6)
text(-30,1,substitute(paste(nv,' points greater than ',10^{-6}),list(nv=nval)),adj=0)
```



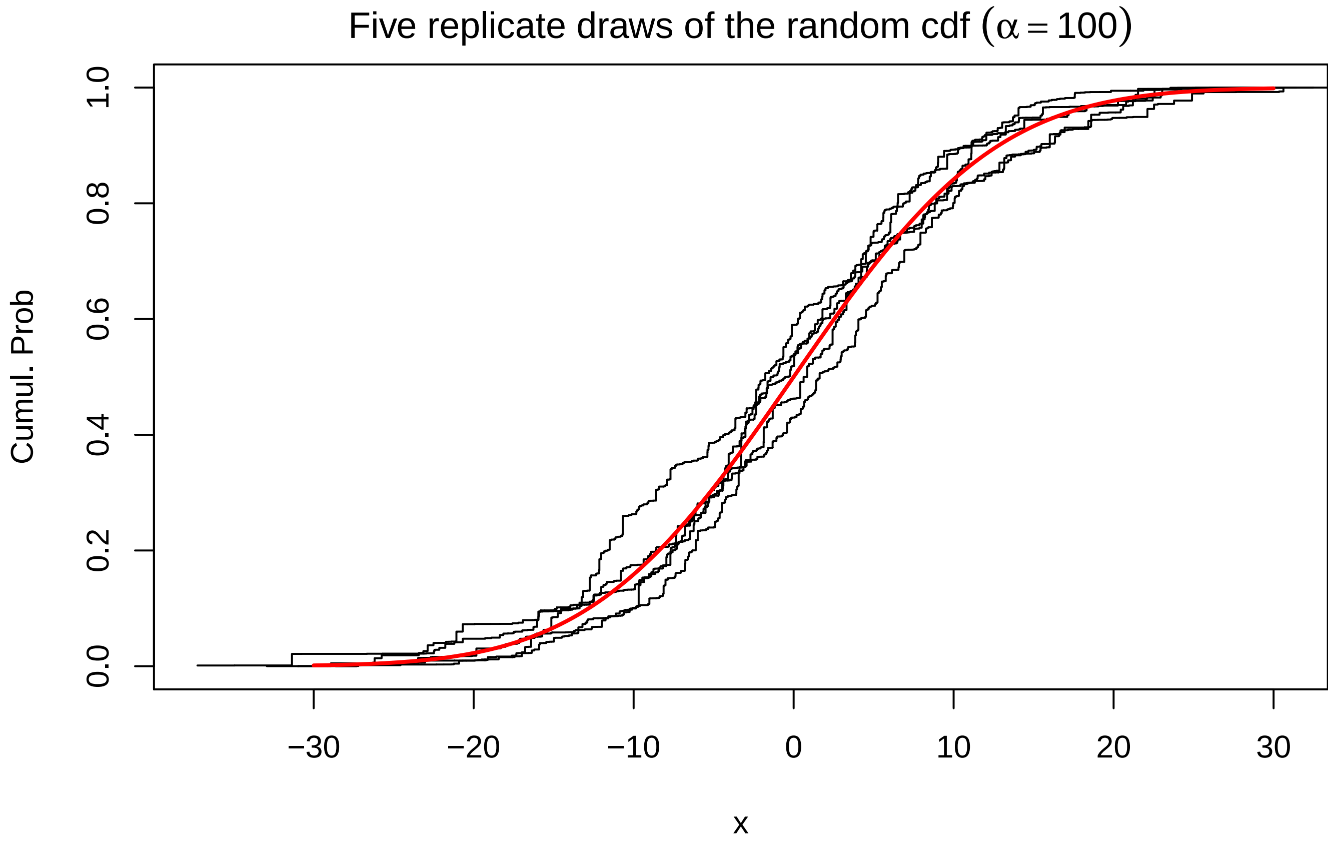
```
Xi<-rnorm(M,0,10)
V<-rbeta(M,1,alpha)
pi.vec<-c(V[1],cumprod(1-V[-M])*V[-1])
Xi2<-Xi
pi2<-pi.vec

Xi<-rnorm(M,0,10)
V<-rbeta(M,1,alpha)
W<-1-V
pi.vec<-c(V[1],cumprod(1-V[-M])*V[-1])
Xi3<-Xi
pi3<-pi.vec

par(mar=c(4,4,2,0))
plot(Xi1,pi1,type="n",ylim=range(0,1),ylab=expression(pi),xlim=range(-30,30))
title(expression(paste('Three draws from the ',DP(alpha,G[X]))))
points(Xi1[pi1>1e-6],pi1[pi1>1e-6],pch=19,xlim=range(-30,30),cex=0.5)
for(i in 1:M){
        if(pi1[i] < 1e-6) next
        lines(c(Xi1[i],Xi1[i]),c(0,pi1[i]),lty=2)
}
points(Xi2[pi2>1e-6],pi2[pi2>1e-6],pch=19,xlim=range(-30,30),col="red",cex=0.5)
for(i in 1:M){
        if(pi2[i] < 1e-6) next
        lines(c(Xi2[i],Xi2[i]),c(0,pi2[i]),lty=2,col="red")
}
points(Xi3[pi3>1e-6],pi3[pi3>1e-6],pch=19,xlim=range(-30,30),col="green",cex=0.5)
for(i in 1:M){
        if(pi3[i] < 1e-6) next
        lines(c(Xi3[i],Xi3[i]),c(0,pi3[i]),lty=2,col="green")
}
```



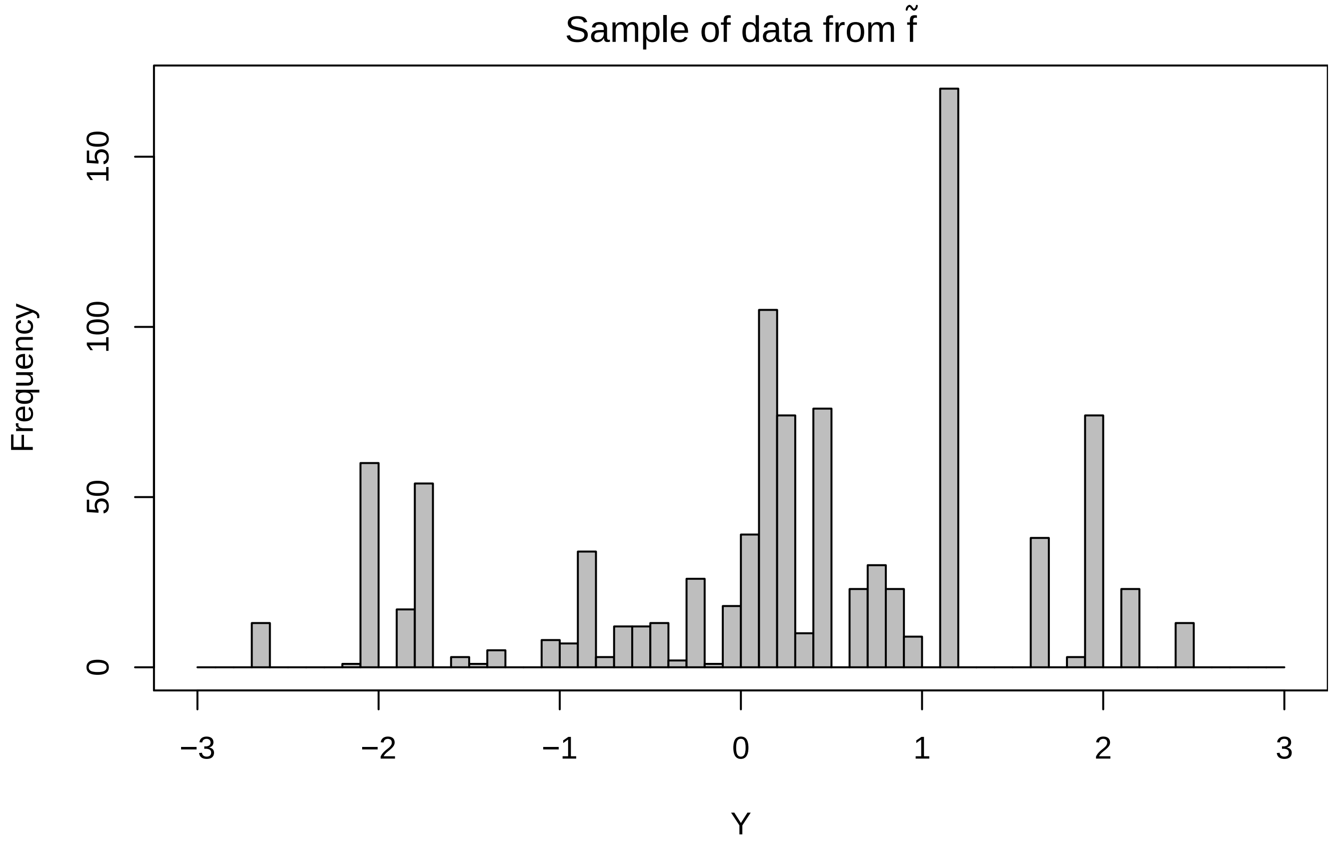
```
Xi<-rnorm(M,0,10)
V<-rbeta(M,1,alpha)
pi.vec<-c(V[1],cumprod(1-V[-M])*V[-1])
XiLarge<-Xi
piLarge<-pi.vec

xi.sort<-XiLarge[order(XiLarge)]
pi.sort<-pi.vec[order(XiLarge)]
Pi<-cumsum(pi.sort)
par(mar=c(4,4,2,0))
plot(xi.sort,Pi,type="s",ylim=range(0,1),xlab="x",ylab="Cumul. Prob")
title(expression(paste('Five replicate draws of the random cdf ',(alpha==2))))
for(i in 1:4){
        Xi<-rnorm(M,0,10)
        V<-rbeta(M,1,alpha)
    pi.vec<-c(V[1],cumprod(1-V[-M])*V[-1])
        XiLarge<-Xi
        piLarge<-pi.vec
        xi.sort<-XiLarge[order(XiLarge)]
        pi.sort<-pi.vec[order(XiLarge)]
        Pi<-cumsum(pi.sort)
        lines(xi.sort,Pi,type="s")
}
xv<-seq(-30,30,length=1001)
yv<-pnorm(xv,0,10)
lines(xv,yv,col="red",lwd=2)
```



The parameter α is a precision parameter; as α increases, the random distribution becomes more concentrated on the base measure.



The random distribution $\widetilde{f}$ can be sampled to produce iid data $Y_1, \ldots, Y_n$ in a straightforward fashion. In the simulation below, we simulate $\widetilde{f}$ with $\alpha = 10$ and $G_X \equiv Normal(0,1)$



Note that in the sample of size $n = 1000$, there are only 64 distinct Y values

```
tabY<-table(Y)
names(tabY)<-as.character(round(as.numeric(names(tabY)),5))
tabY

+ -2.63151 -2.16276 -2.08518 -1.81156 -1.75768 -1.70421 -1.54031 -1.40852
+       13        1       60       17       45        9        3        1
+ -1.32673 -1.09304 -0.90964 -0.86634 -0.83135 -0.78091 -0.73734 -0.69322
+        5        8        7        1       33        2        1        1
+ -0.66846 -0.62998 -0.53459 -0.50782 -0.47949 -0.41675 -0.33048 -0.32752
+        8        3       11        1       12        1        1        1
+ -0.29525 -0.28987 -0.26925  -0.1323 -0.09247 -0.08415  -0.0704 -0.05806
+        1        4       21        1        1        6        5        5
+  -0.0233  0.01556  0.10959  0.18201  0.21553   0.2375  0.25179  0.25219
+        1       39      104        1        4       13        1        8
+  0.25484  0.28758  0.31203   0.3336   0.3526  0.38624  0.41278  0.49196
+        1       47        5        1        3        1       69        7
+  0.60717  0.61285  0.65872  0.70102  0.71598  0.72779  0.81569  0.94265
+       11        9        3        2        2       26       23        9
+  1.11499  1.13418  1.60025  1.68618  1.85047  1.99737  2.13939  2.43685
+      132       38        3       35        3       74       23       13
```

and thus we observe *clustering*: this is due to the fact that $\widetilde{f}$ is discrete.

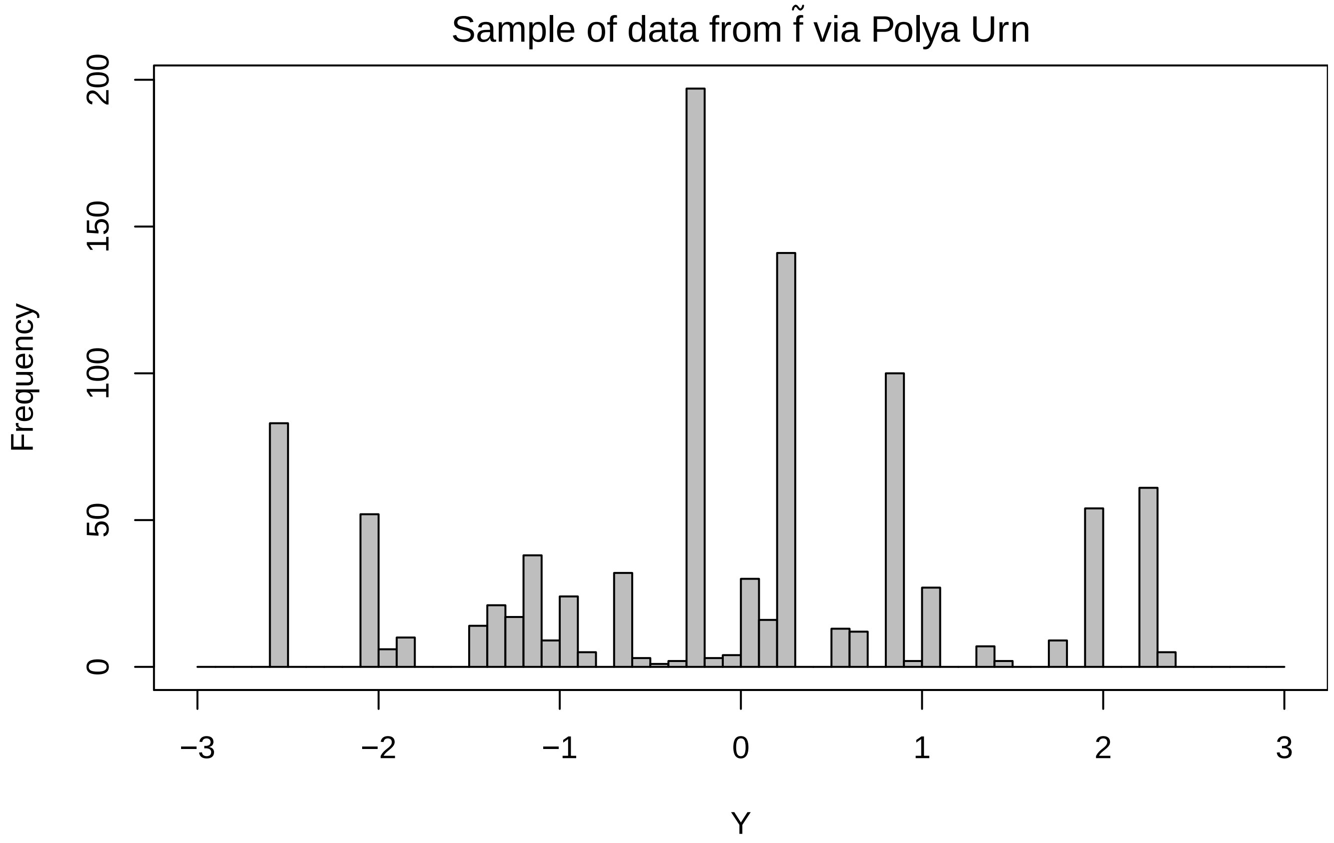
We can also use a *Polya Urn* scheme to sample these data: this recursive scheme proceeds as follows:

1. simulate $Y_1 \sim G_X$;
2. for $i = 2, 3, \ldots, n$ generate Y_i from the mixture distribution

$$\frac{\alpha}{\alpha + i - 1} G_X(.) + \frac{1}{\alpha + i - 1} \sum_{j=1}^{i-1} \delta_{Y_j}(.)$$

   that is

   - with probability $\alpha/(\alpha + i - 1)$, simulate $Y_i \sim G_X(.)$;
   - with probability $1/(\alpha + i - 1)$, simulate Y_i uniformly on $\{Y_1, \ldots, Y_{i-1}\}$.



Here there are 49 distinct Y values. The Polya Urn method reconstructs a sample of exchangeable observables according to the de Finetti representation

$$f_{Y_1,\ldots,Y_n}(y_1, \ldots, y_n) = \int \prod_{i=1}^{n} \widetilde{f}(y_i)\, \pi_0(d\widetilde{f})$$

by sampling directly from the left hand side using the factorization

$$f_{Y_1,\ldots,Y_n}(y_1, \ldots, y_n) = f_{Y_1}(y_1) \prod_{i=2}^{n} f_{Y_i|Y_1,\ldots,Y_{i-1}}(y_i|y_1, \ldots, y_{i-1})$$

where

$$f_{Y_i|Y_1,\ldots,Y_{i-1}}(y_i|y_1, \ldots, y_{i-1}) = \frac{\alpha}{\alpha + i - 1} g_X(y_i) + \frac{1}{\alpha + i - 1} \sum_{j=1}^{i-1} \delta_{Y_j}(y_i)$$

where $g_X(.)$ is the density corresponding to $G_X(.)$.

In a *Dirichlet Process Mixture* model we add another stage that brings in a continuous distribution. For example, could treat each x_i as the location of a normal density, and consider generating a y for each

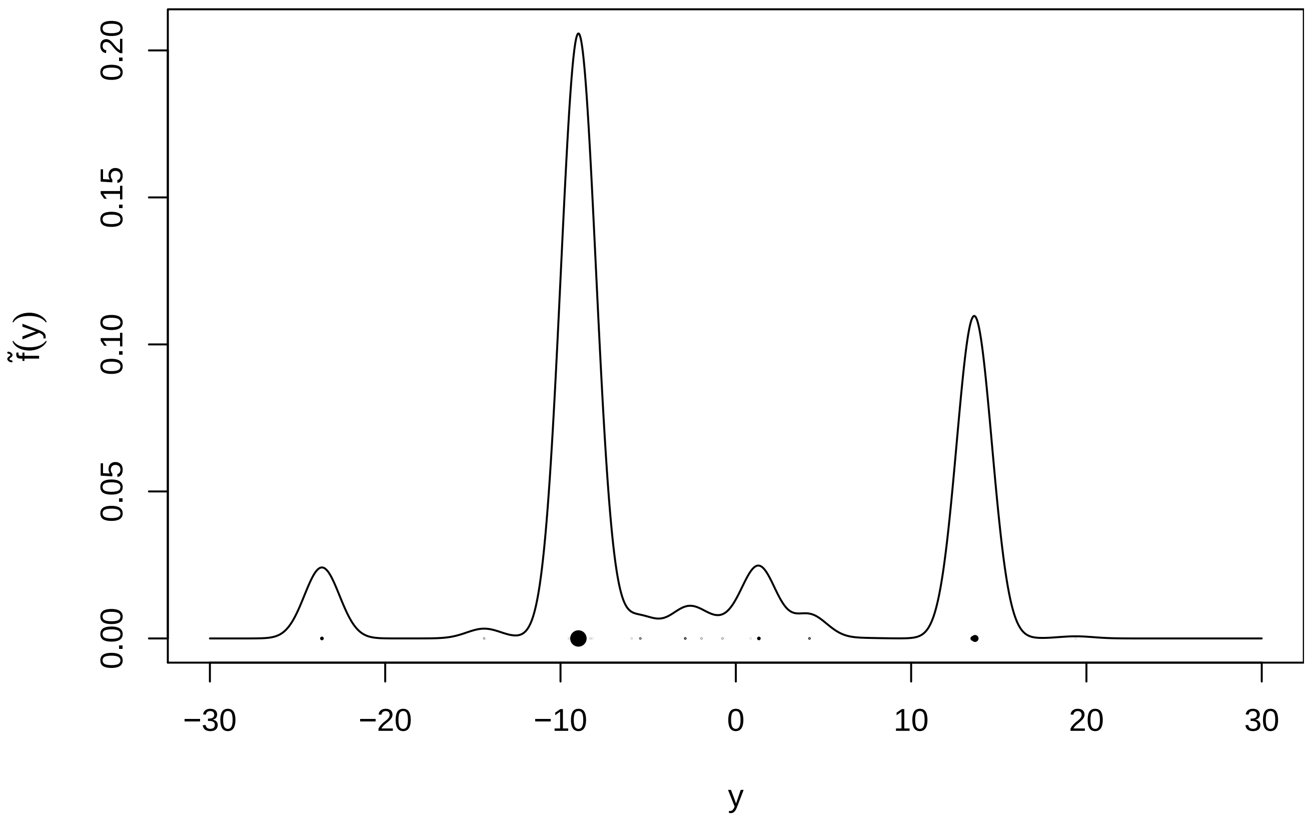
$$\begin{array}{rcll} x_1, x_2, \ldots & \sim & G_X & \\ \pi_1, \pi_2, \ldots & & \text{generated by stick-breaking.} & \\ y_i & \sim & \phi((y_i - x_i)/\sigma) & \quad i = 1, 2, \ldots \end{array}$$

Then the random density function generated takes the form

$$\widetilde{f}(y) = \sum_{i=1}^{\infty} \pi_i \phi((y - x_i)/\sigma)/\sigma$$

that is, an **infinite mixture model**. In the plot below we have $\alpha = 2$, with $G_X(x) \equiv Normal(0, \lambda^2)$ with $\lambda = 10$.

```
set.seed(1010)
M<-1001
alpha<-2
sigma<-1;lambda<-10
Xi<-rnorm(M,0,lambda)
V<-rbeta(M,1,alpha)
pi.vec<-c(V[1],cumprod(1-V[-M])*V[-1])
xv<-seq(-30,30,by=0.01)
yv<-xv*0
for(i in 1:M){
    yv<-yv+pi.vec[i]*dnorm(xv,Xi[i],sigma)
}
par(mar=c(4,5,2,0))
plot(xv,yv,type='l',xlab='y',ylab=expression(widetilde(f)(y)))
points(Xi,Xi*0,cex=2*pi.vec,pch=19)
```



Gibbs sampler: For fully Bayesian inference consider the de Finetti construction

- a prior model for f that is $DPM(\alpha, G_X, g_Y; \theta)$ where θ represents the parameters that appear in G_X and g_Y.
- conditional on f, data $y_1, y_2, \ldots, y_n \sim f$ independently

We wish to compute the posterior for f. We use the hierarchical formulation

$$\begin{aligned} y_j | x_j &\overset{\text{ind.}}{\sim} g_Y(y_j | x_j; \theta) \qquad j = 1, \ldots, n \\ x_1, \ldots, x_n &\sim DP(\alpha, G_X; \theta) \\ \theta &\sim \pi_0(\theta). \end{aligned}$$

The latent variables $x_1, \ldots, x_n$ are also treated as parameters. They can be sampled using an MCMC **Gibbs sampler** scheme. For $j = 1, \ldots, n$, we sample

$$x_j \mid \boldsymbol{x}_{(j)}, \boldsymbol{y} \sim w_0 p(x_j | y_j) + \sum_{l \neq j} w_l \delta_{x_l}$$

where

- $\boldsymbol{y} = (y_1, \ldots, y_n)^\mathsf{T}$
- $\boldsymbol{x}_{(j)} = (x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n)^\mathsf{T}$.
- w_0 is proportional to α times the **prior predictive** of y_j
- w_l is proportional to the **likelihood** of $y_j | x_l$
- $p(x_j | y_j)$ is the **posterior** for x_j given y_j.

Marginalized sampler: To be more efficient, we can use the **clustering** property: suppose that at a given iteration of the MCMC, there are K clusters labelled 1 to K, where $K \leq n$. Label the K distinct x values $z_1, \ldots, z_K$ and for each j, define the corresponding cluster label c_j where

$$c_j = k \qquad \Longleftrightarrow \qquad x_j = z_k$$

We can update the c_js instead of the x_js which will be more computationally efficient; we are clustering xs to the **cluster centres** at the z values. For $i = 1, \ldots, n$, let

- $n_1(i), \ldots, n_K(i)$ denote the number of items in clusters $1, \ldots, K$
- $\boldsymbol{y}_1(i), \ldots, \boldsymbol{y}_K(i)$ denote the vectors of y values currently allocated to the K clusters

if the ith data point is removed. For $i = 1, \ldots, n$, we sample the cluster labels in a Gibbs sampler with conditional probabilities

$$\Pr[c_i = k \mid c_{(i)}] \propto \frac{n_k(i)}{\alpha + n - 1} p(y_i | \boldsymbol{y}_k(i)) \qquad k = 1, \ldots, K$$

and

$$\Pr[c_i = K + 1 \mid c_{(i)}] \propto \frac{\alpha}{\alpha + n - 1} p(y_i)$$

In this expression

- $p(y_i | \boldsymbol{y}_k(i))$ is the **posterior predictive** density for y_i, assuming that y_i comes from cluster k.
- $p(y_i)$ is the **prior predictive** density for y_i, assuming that y_i comes from a **new cluster** not currently represented in the data.

In this formulation, we have **integrated out** the Dirichlet Process analytically. Thus we can simply sample the cluster labels in turn, and then sample the $z_1, \ldots, z_k$ values; this will allow us to do density estimation. By the usual calculation

$$p(y_i | \boldsymbol{y}_k(i)) = \int g_Y(y_i \mid x) p(x \mid \boldsymbol{y}_k(i))\, dx$$

where

$$p(x \mid \boldsymbol{y}_k(i)) \quad \propto \quad p(\boldsymbol{y}_k(i) \mid x) p(x) = \left\{ \prod_{l \neq i} g_Y(y_l \mid x) \right\} p(x)$$

gives the posterior distribution for the kth cluster centre. Similarly

$$p(y_i) = \int g_Y(y_i \mid x)p(x)\, dx$$

In the earlier Normal model, suppose for simplicity that G_X is the $Normal(0, \lambda^2)$ density:

$$x_i \sim Normal(0, \lambda^2) \qquad\qquad y_i \mid x_i \sim Normal(x_i, \sigma^2)$$

Then by standard calculations

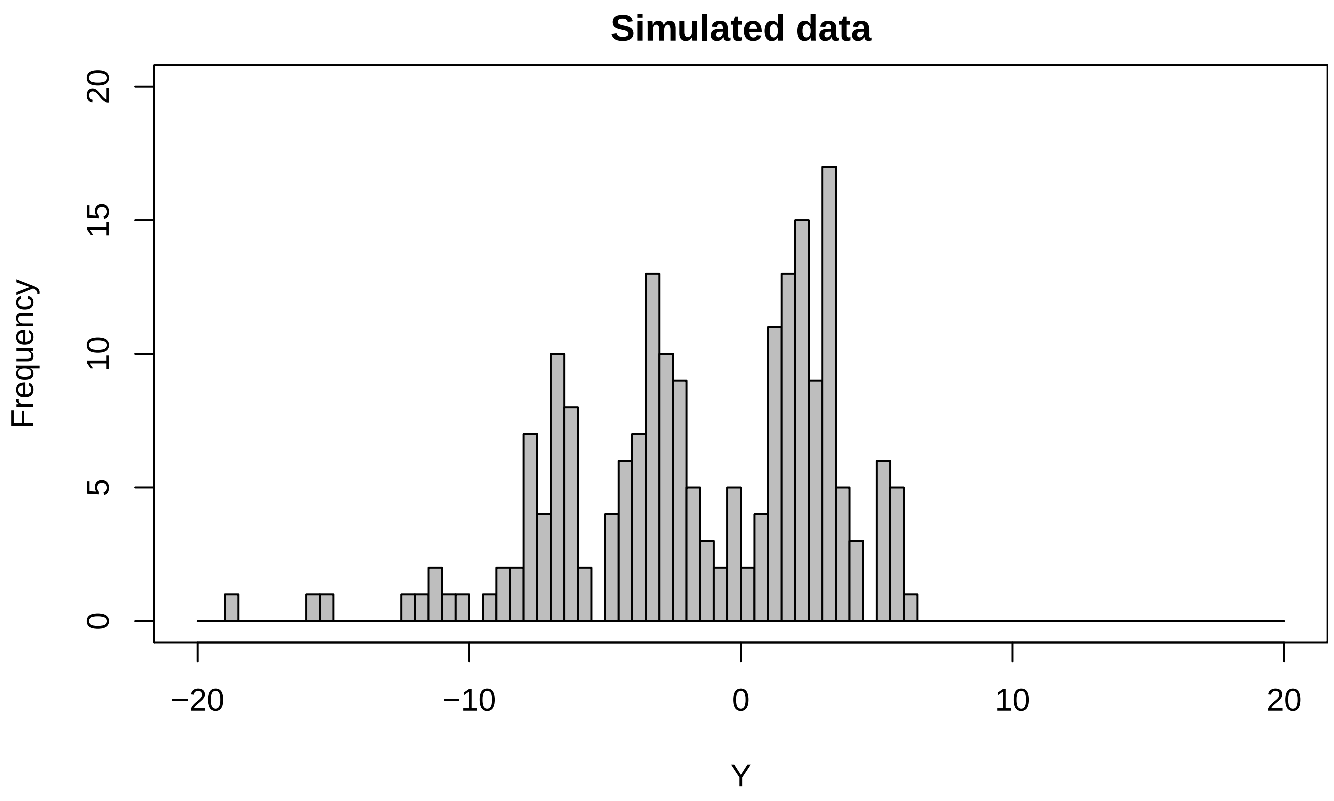
$$p(y_i|\boldsymbol{y}_k(i)) \equiv Normal\left(\frac{n_k(i)\overline{y}_k(i)/\sigma^2}{n_k(i)/\sigma^2 + 1/\lambda^2}, \frac{(n_k(i)+1)/\sigma^2 + 1/\lambda^2}{n_k(i)/\sigma^2 + 1/\lambda^2}\sigma^2\right)$$

and

$$p(y_i) \equiv Normal\left(0, \sigma^2 + \lambda^2\right).$$

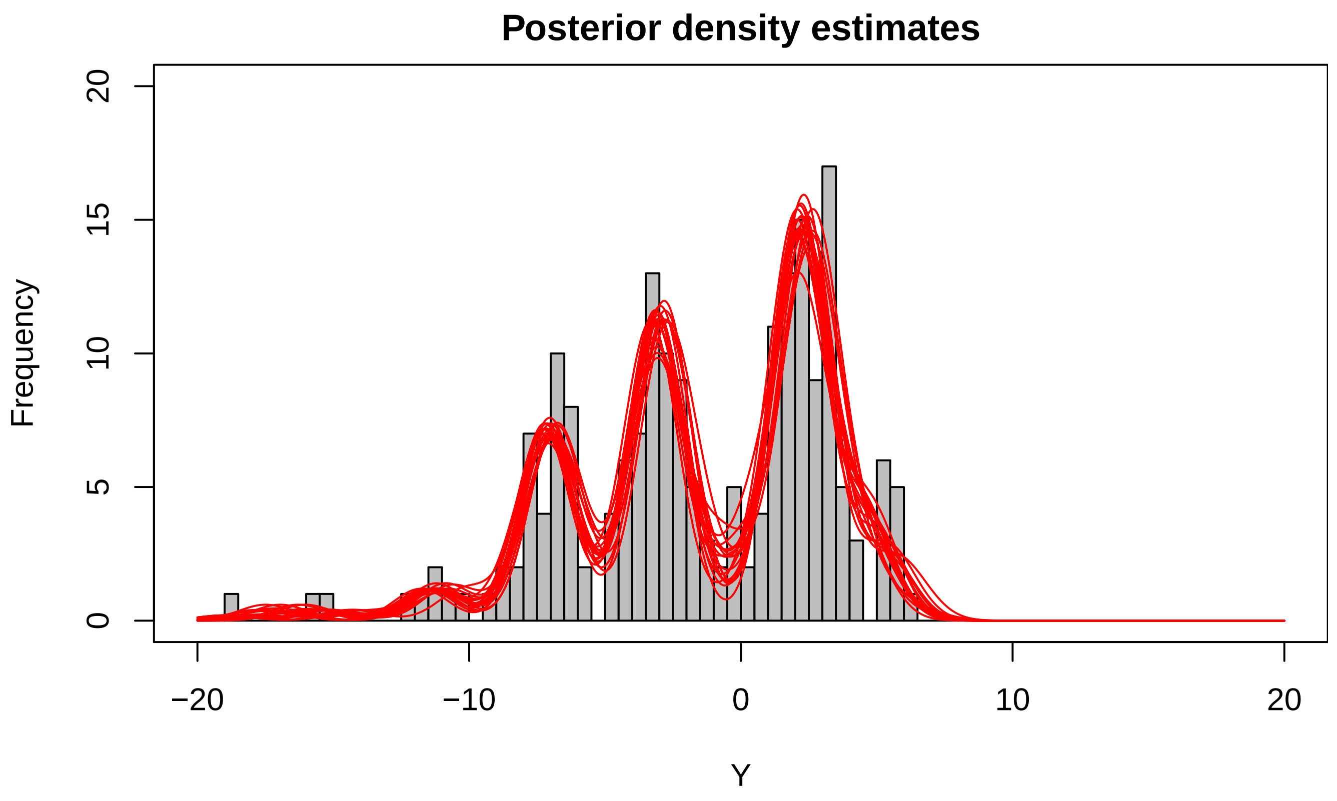
The marginalized version of the algorithm often mixes more quickly than the Gibbs sampler on the cluster centres.

```
set.seed(1091)
n<-200
alpha<-2
sigma<-1; lambda<-10
X<-numeric(n)
X[1]<-rnorm(1,0,lambda)
for(j in 2:n){
        u<-runif(1)
        if(u < alpha/(alpha+j-1)){
                X[j]<-rnorm(1,0,lambda)
        }else{
                X[j]<-sample(X[1:(j-1)],size=1)
        }
}
Y<-rnorm(n,X,sigma)
par(mar=c(4,4,2,0))
hist(Y,breaks=seq(-20,20,by=0.50),ylim=range(0,20),col='gray',main='Simulated data');box()
```

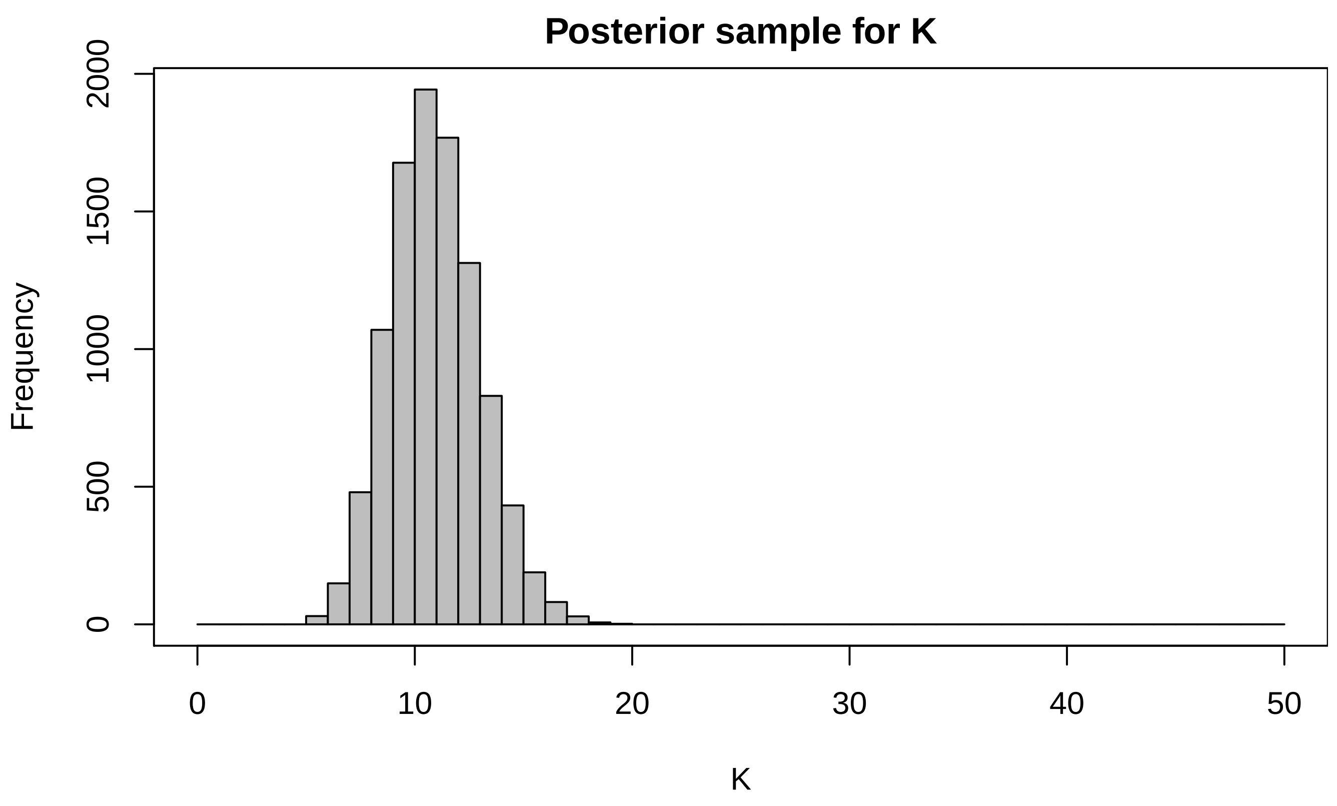


```r
True.K<-length(table(X))

#####################################################
set.seed(1010)
index<-c(1:n)
Kval<-5
Yq<-quantile(Y,prob=c(0:Kval)/Kval)
Cvec<-as.numeric(cut(Y,Yq,include.lowest=T))
Ysum.Vec<-rep(0,Kval)
for(k in 1:Kval){Ysum.Vec[k]<-sum(Y[Cvec==k])}
Nvec<-as.numeric(table(Cvec))

nreps<-10000
Nsamp<-rep(0,nreps)
yv<-seq(-20,20,by=0.01)
fyv.samp<-numeric()

for(irep in 1:nreps){

        for(i in 1:n){
                sum.Vec<-Ysum.Vec
                sum.Vec[Cvec[i]]<-sum.Vec[Cvec[i]]-Y[i]
                nvec<-Nvec
                nvec[Cvec[i]]<-nvec[Cvec[i]]-1
                if(nvec[Cvec[i]] > 0){
                        mvec<-sum.Vec/nvec
                        Mvec<-(sum.Vec/sigma^2)/(nvec/sigma^2+1/lambda^2)
                        Svec<-sigma^2*((nvec+1)/sigma^2+1/lambda^2)/(nvec/sigma^2+1/lambda^2)
                        pvec<-nvec*dnorm(rep(Y[i],Kval),Mvec,sqrt(Svec))
                        pvec<-c(pvec,alpha*dnorm(Y[i],0,sqrt(sigma^2 + lambda^2)))/(n-1+alpha)
                        pvec<-pvec/sum(pvec)
                        Ksamp<-sample(c(1:(Kval+1)),size=1,prob=pvec)
                        Cvec[i]<-Ksamp
                        if(Ksamp == Kval+1){
                                Kval<-Kval+1
                                Ysum.Vec<-c(sum.Vec,Y[i])
                                Nvec<-c(nvec,1)
                        }else{
                                Ysum.Vec<-sum.Vec
                                Ysum.Vec[Ksamp]<-sum.Vec[Ksamp]+Y[i]
                                Nvec<-nvec
                                Nvec[Ksamp]<-nvec[Ksamp]+1
                        }
                }else{
                        Ktmp<-Cvec[i]
                        sum.Vec<-sum.Vec[-Ktmp]
                        nvec<-nvec[-Ktmp]
                        Kval<-Kval-1
                        Cvec[Cvec > Ktmp]<-Cvec[Cvec > Ktmp]-1

                        mvec<-sum.Vec/nvec
                        Mvec<-(sum.Vec/sigma^2)/(nvec/sigma^2+1/lambda^2)
                        Svec<-sigma^2*((nvec+1)/sigma^2+1/lambda^2)/(nvec/sigma^2+1/lambda^2)
                        pvec<-nvec*dnorm(rep(Y[i],Kval),Mvec,sqrt(Svec))
                        pvec<-c(pvec,alpha*dnorm(Y[i],0,sqrt(sigma^2 + lambda^2)))/(n-1+alpha)

                        pvec<-pvec/sum(pvec)
                        Ksamp<-sample(c(1:(Kval+1)),size=1,prob=pvec)
                        Cvec[i]<-Ksamp
                        if(Ksamp == Kval+1){
                                Kval<-Kval+1
                                Ysum.Vec<-c(sum.Vec,Y[i])
                                Nvec<-c(nvec,1)
                        }else{
                                Ysum.Vec<-sum.Vec
                                Ysum.Vec[Ksamp]<-sum.Vec[Ksamp]+Y[i]
                                Nvec<-nvec
                                Nvec[Ksamp]<-nvec[Ksamp]+1
                        }
                }
        }


        Nsamp[irep]<-Kval

        if(irep %% 500 == 0){

                MuVec<-(Ysum.Vec/sigma^2)/(Nvec/sigma^2+1/lambda^2)
                SigVec<-1/(Nvec/sigma^2+1/lambda^2)
                Xvec<-rnorm(Kval)*sqrt(SigVec)+MuVec
                fyv<-yv*0
                for(k in 1:Kval){
                        fyv<-fyv+Nvec[k]*dnorm(yv,Xvec[k],sigma)/n
                }
                fyv.samp<-cbind(fyv.samp,fyv)
        }
}
```

```
par(mar=c(4,4,2,0))
hist(Y,breaks=seq(-20,20,by=0.5),col='gray',ylim=range(0,20),main='Posterior density estimates');box()
for(i in 1:ncol(fyv.samp)){lines(yv,fyv.samp[,i]*n*0.5,col='red')}
```



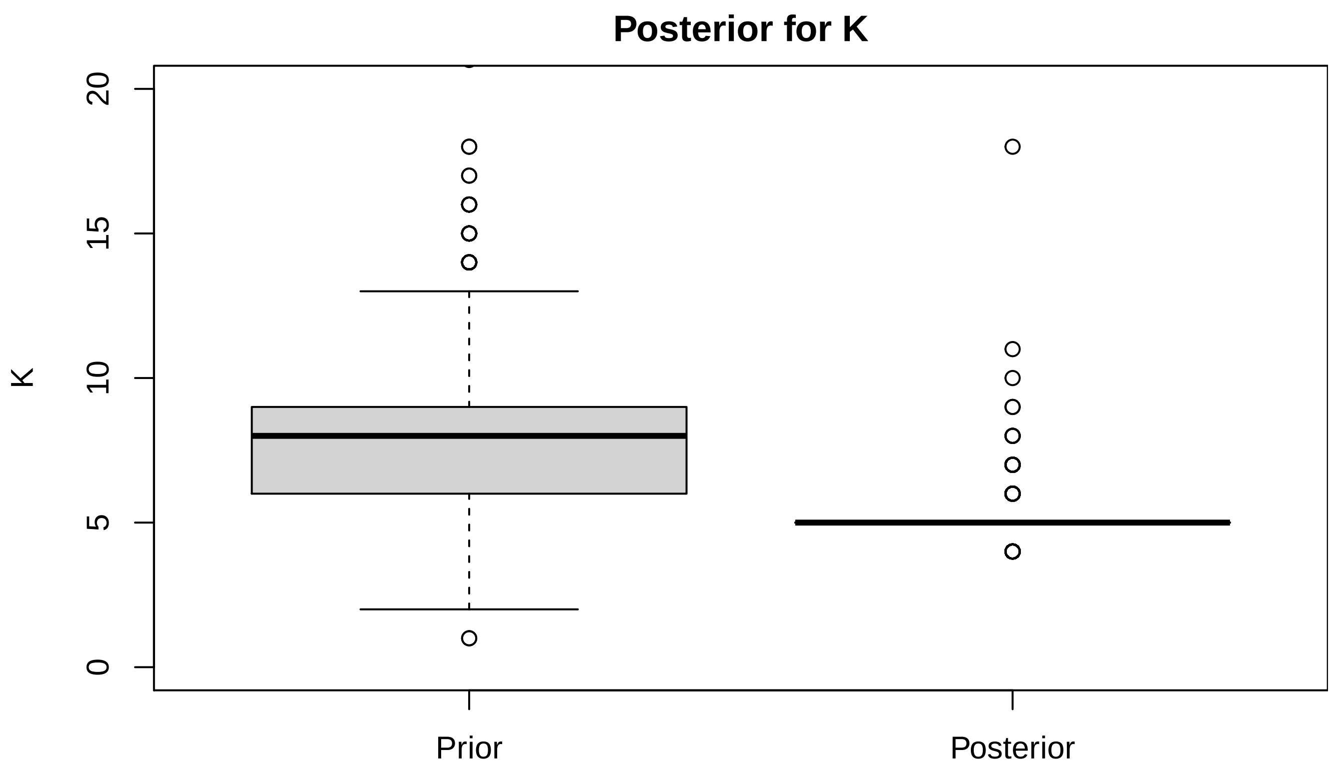
```r
hist(Nsamp,col='gray',breaks=seq(0,50,by=1),xlab='K',main='Posterior sample for K');box()
```

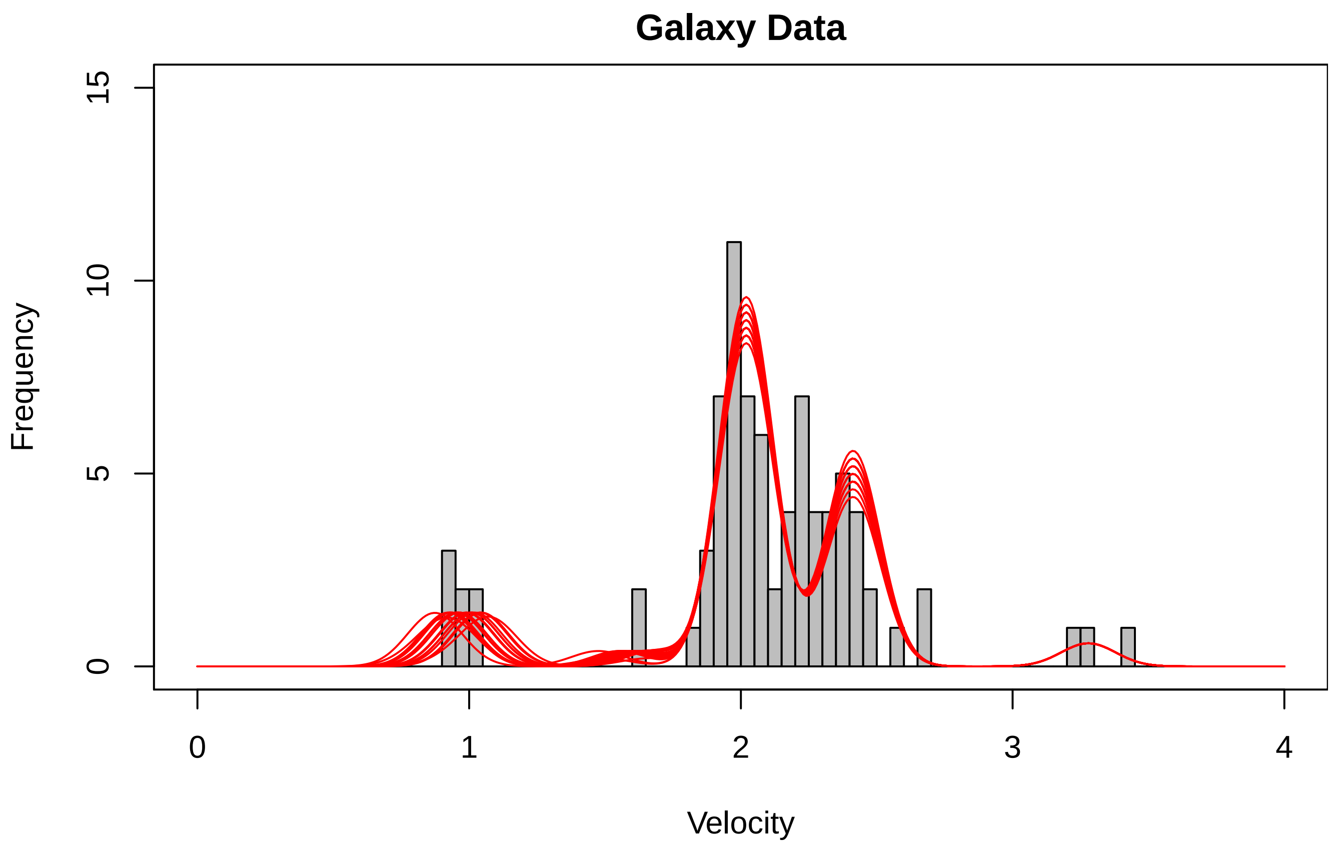


```r
#Galaxy data using Gibbs sampler
Y.galaxy<-MASS::galaxies/10000
n.galaxy<-length(Y.galaxy)

set.seed(1010)
index<-c(1:n.galaxy)
nreps<-10000

alpha<-2.0

Pi<-Y.galaxy
Prior.K.galaxy<-numeric()

for(irep in 1:nreps){
        for(i in 1:n.galaxy){
                pvec<-rep(1/(alpha+n.galaxy-1),n.galaxy)
                pvec[i]<-alpha/(alpha+n.galaxy-1)
                ival<-sample(index,size=1,prob=pvec)
                if(ival == i){
                        Pi[i]<-rnorm(1,0,lambda)
                }else{
                        Pi[i]<-Pi[ival]
                }
        }

        Kval<-length(table(Pi))
        Prior.K.galaxy<-c(Prior.K.galaxy,Kval)
}

set.seed(1010)
```

```
index<-c(1:n.galaxy)
nreps<-10000
sigma<-0.1
lambda<-0.5

Pi<-Y.galaxy
Pvec<-rep(0,n.galaxy)
Posterior.K.galaxy<-numeric()
yv<-seq(0,4,by=0.01)
fyv.samp.GAL<-numeric()
for(irep in 1:nreps){

        for(i in 1:n.galaxy){
                pvec<-dnorm(Y.galaxy[i],Pi,rep(sigma,n.galaxy))
                pvec[i]<-alpha*dnorm(Y.galaxy[i],0,sqrt(sigma^2+lambda^2))
                ival<-sample(index,size=1,prob=pvec)
                if(ival == i){
                        muval<-(Y.galaxy[i]/sigma^2)/(1/sigma^2+1/lambda^2)
                        sval<-sqrt(1/(1/sigma^2+1/lambda^2))
                        Pi[i]<-rnorm(1,muval,sval)
                }else{
                        Pi[i]<-Pi[ival]
                }
        }

        Kval<-length(table(Pi))
    Posterior.K.galaxy<-c(Posterior.K.galaxy,Kval)

        if(irep %% 100 == 0){
                fyv<-yv*0
                for(k in 1:n.galaxy){
                        fyv<-fyv+dnorm(yv,Pi[k],sigma)/n.galaxy
                }
                fyv.samp.GAL<-cbind(fyv.samp.GAL,fyv)
        }

}
```

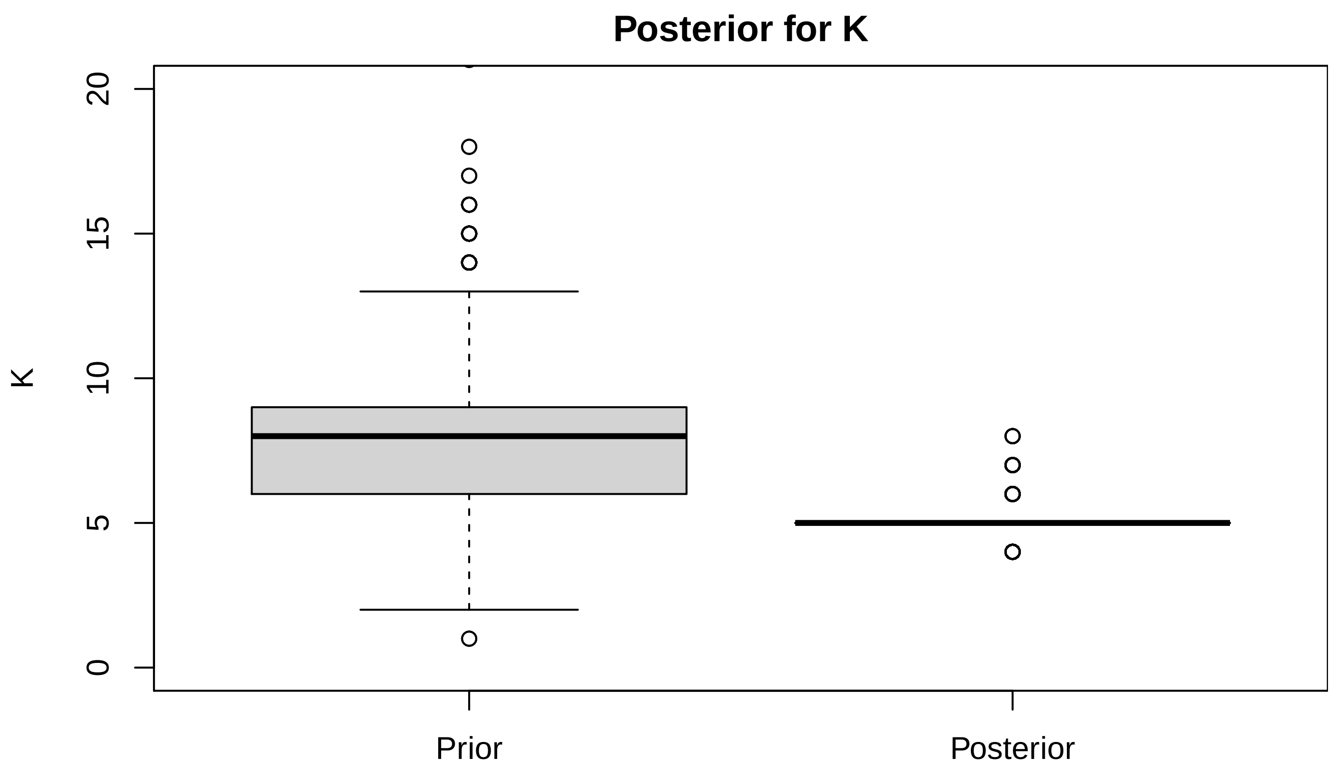
```
table(Posterior.K.galaxy)/length(Posterior.K.galaxy)

+ Posterior.K.galaxy
+      4      5      6      7      8      9     10     11     18     28
+ 0.0512 0.8120 0.1260 0.0096 0.0006 0.0002 0.0001 0.0001 0.0001 0.0001

table(Prior.K.galaxy)/length(Prior.K.galaxy)

+ Prior.K.galaxy
+      1      2      3      4      5      6      7      8      9     10     11
+ 0.0002 0.0019 0.0135 0.0382 0.0862 0.1399 0.1638 0.1686 0.1483 0.1072 0.0651
+     12     13     14     15     16     17     18     21     28
+ 0.0383 0.0179 0.0062 0.0032 0.0009 0.0002 0.0002 0.0001 0.0001
```

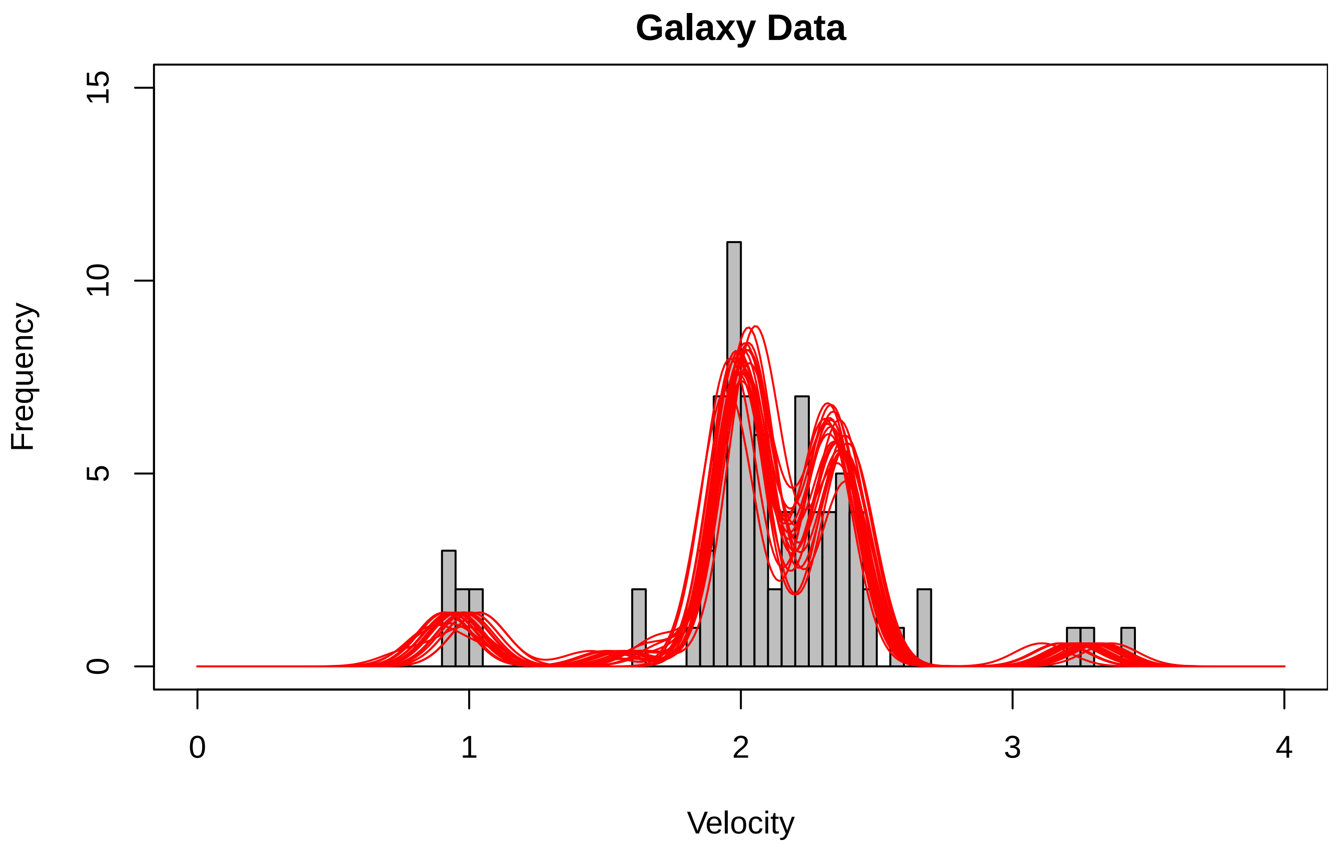

Posterior for K
K
0
5
10
15
20
Prior
Posterior


Galaxy Data
Frequency
0
5
10
15
0
1
2
3
4
Velocity

```
#Galaxy data using marginalized algorithm
Y.galaxy<-MASS::galaxies/10000
n.galaxy<-length(Y.galaxy)

set.seed(1010)
index<-c(1:n.galaxy)
nreps<-10000
alpha<-2.0
sigma<-0.1
lambda<-0.5

index<-c(1:n.galaxy)
Kval<-5
Yq<-quantile(Y.galaxy,prob=c(0:Kval)/Kval)
Cvec<-as.numeric(cut(Y.galaxy,Yq,include.lowest=T))
Ysum.Vec<-rep(0,Kval)
for(k in 1:Kval){Ysum.Vec[k]<-sum(Y.galaxy[Cvec==k])}
Nvec<-as.numeric(table(Cvec))

Posterior.K.marg<-rep(0,nreps)
yv<-seq(0,4,by=0.01)
marg.fyv.samp.GAL<-numeric()

for(irep in 1:nreps){

        for(i in 1:n.galaxy){
                sum.Vec<-Ysum.Vec
                sum.Vec[Cvec[i]]<-sum.Vec[Cvec[i]]-Y.galaxy[i]
                nvec<-Nvec
                nvec[Cvec[i]]<-nvec[Cvec[i]]-1
                if(nvec[Cvec[i]] > 0){
                        mvec<-sum.Vec/nvec
                        Mvec<-(sum.Vec/sigma^2)/(nvec/sigma^2+1/lambda^2)
                        Svec<-sigma^2*((nvec+1)/sigma^2+1/lambda^2)/(nvec/sigma^2+1/lambda^2)
                        pvec<-nvec*dnorm(rep(Y.galaxy[i],Kval),Mvec,sqrt(Svec))
                        pvec<-c(pvec,alpha*dnorm(Y.galaxy[i],0,sqrt(sigma^2 + lambda^2)))
                        pvec<-pvec/sum(pvec)
                        Ksamp<-sample(c(1:(Kval+1)),size=1,prob=pvec)
                        Cvec[i]<-Ksamp
                        if(Ksamp == Kval+1){
                                Kval<-Kval+1
                                Ysum.Vec<-c(sum.Vec,Y.galaxy[i])
                                Nvec<-c(nvec,1)
                        }else{
                                Ysum.Vec<-sum.Vec
                                Ysum.Vec[Ksamp]<-sum.Vec[Ksamp]+Y.galaxy[i]
                                Nvec<-nvec
                                Nvec[Ksamp]<-nvec[Ksamp]+1
                        }
                }else{
                        Ktmp<-Cvec[i]
                        sum.Vec<-sum.Vec[-Ktmp]
                        nvec<-nvec[-Ktmp]
                        Kval<-Kval-1
                        Cvec[Cvec > Ktmp]<-Cvec[Cvec > Ktmp]-1

                        mvec<-sum.Vec/nvec
                        Mvec<-(sum.Vec/sigma^2)/(nvec/sigma^2+1/lambda^2)
                        Svec<-sigma^2*((nvec+1)/sigma^2+1/lambda^2)/(nvec/sigma^2+1/lambda^2)
                        pvec<-nvec*dnorm(rep(Y.galaxy[i],Kval),Mvec,sqrt(Svec))
                        pvec<-c(pvec,alpha*dnorm(Y.galaxy[i],0,sqrt(sigma^2 + lambda^2)))/(n.galaxy-1+alpha
                        pvec<-pvec/sum(pvec)
                        Ksamp<-sample(c(1:(Kval+1)),size=1,prob=pvec)
                        Cvec[i]<-Ksamp
                        if(Ksamp == Kval+1){
                                Kval<-Kval+1
                                Ysum.Vec<-c(sum.Vec,Y.galaxy[i])
                                Nvec<-c(nvec,1)
                        }else{
                                Ysum.Vec<-sum.Vec
                                Ysum.Vec[Ksamp]<-sum.Vec[Ksamp]+Y.galaxy[i]
                                Nvec<-nvec
                                Nvec[Ksamp]<-nvec[Ksamp]+1
                        }
                }
        }


        Posterior.K.marg[irep]<-Kval

        if(irep %% 100 == 0){

                MuVec<-(Ysum.Vec/sigma^2)/(Nvec/sigma^2+1/lambda^2)
                SigVec<-1/(Nvec/sigma^2+1/lambda^2)
                Xvec<-rnorm(Kval)*sqrt(SigVec)+MuVec
                fyv<-yv*0
                for(k in 1:Kval){
                        fyv<-fyv+Nvec[k]*dnorm(yv,Xvec[k],sigma)/n.galaxy
                }
                marg.fyv.samp.GAL<-cbind(marg.fyv.samp.GAL,fyv)
        }
}
```

```
table(Posterior.K.marg)/nreps

+ Posterior.K.marg
+      4      5      6      7      8
+ 0.2340 0.6559 0.1021 0.0075 0.0005

table(Prior.K.galaxy)/length(Prior.K.galaxy)

+ Prior.K.galaxy
+      1      2      3      4      5      6      7      8      9     10     11
+ 0.0002 0.0019 0.0135 0.0382 0.0862 0.1399 0.1638 0.1686 0.1483 0.1072 0.0651
+     12     13     14     15     16     17     18     21     28
+ 0.0383 0.0179 0.0062 0.0032 0.0009 0.0002 0.0002 0.0001 0.0001
```


Posterior for K
K
0
5
10
15
20
Prior
Posterior


Galaxy Data
Frequency
0
5
10
15
0
1
2
3
4
Velocity